Journal of Innovative Image Processing (ISSN: 2582-4252)
www.irojournals.com/iroiip/

# Robust Framework for Diagnosing Novel Corona Virus from CT images using Support Vector Binary Classifier

**Alok Kumar[1], Dr. N. Mahendran[2]**

[1]Department of Computer Science and Engineering, Government College of Engineering, Anna University, Dharmapuri, India

[2]Department of Electronics and Communication Engineering, M. Kumarasamy College of Engineering, Anna University, Karur, India

**E-mail:** [1]abesalok@gmail.com, [2]mahe.sec@gmail.com

## Abstract

The novel coronavirus (commonly abbreviated as CoVID-19) has emerged as a threat to the entire global civilization and has emerged as one of the most infectious and, at times, deadly viruses. Prompt discovery of this infection can assist medical supervisors in taking preventive actions to control the spread. Usually, radiologists and medical specialists require an average time of ~31 minutes to test the CT images and confirm the infection. A large dataset of more than 1000 patients has been gathered and randomly chosen for this experiment. In this research, a ready-to-deploy computer-aided diagnosis (CADx) to detect COVID-19 infection is introduced. A variety of deep learning architectures have been experimented to discover the most reliable predictive model for the diagnosis. This research uses the Densely Connected Convolution Network (DenseNet-121 architecture) along with a boosting support vector binary classifier to tell the difference between someone who has the coronavirus and someone who is healthy. The combination put forward in this work achieved 93% ± 1.8% accuracy, 94.9% ± 2.6% recall, 98% ± 1.5% precision, and an F1 score of 94% ± 1.7%. The model takes less than 1 second to process one image. On the grounds of the above findings, it can be concluded that the proposed approach can be used to diagnose novel coronavirus infections.


*DOI: https://doi.org/10.36548/jiip.2023.4.006*
*Received: 17.11.2023, received in revised form: 12.12.2023, accepted: 24.12.2023, published: 03.01.2024*

**Keywords:** Chest CT, DenseNet-121, COVID-19, Support Vector Machine Classifier, Deep Learning.

## 1. Introduction

The World Health Organization (WHO) declared the novel coronavirus, 2019-nCoV, to be an epidemic and Coronavirus Disease 2019 (COVID-19) on February 11, 2020 [1]. It began in Wuhan and rapidly spread to the rest of the world. In March, the World Health Organization (WHO) classified COVID-19 as a pandemic due to its alarming rate of spread and severity. Coronaviruses are a grouping of interconnected RNA viruses that infect mammals with illness. Inflammatory infections induced by them can range in severity from mild to fatal [2].

The first and most essential method for identifying COVID-19 is the polymerase chain reaction (PCR). Nasopharyngeal and oropharyngeal respiratory specimens, among others, can yield SARS-CoV-2 RNA [3]. For the findings, a lot of lab work must be done after sample collection, which takes a lot of time, even if this procedure is the most successful.

A more rapid option is to examine chest radiography images; however, interpreting the small changes requires a professional study. In order to resolve this problem, a number of AI-based algorithms have been developed to detect COVID-19 from radiography images. Furthermore, compared to conventional techniques, which involve radiologists manually reviewing concepts, AI solutions are more rapid. Previous researchers have developed a few AI-based methods for detecting COVID-19, but these methods contain numerous defects.

The diagnosis of the worldwide pandemic and epidemic infection Coronavirus (CoVID-19) is the main topic of this study. There are numerous works by various authors to predict the infection using computed tomography (CT) images, but those works have become unreliable due to the lack of accurate chest CT images. In this research, we gathered about more than 100,000 CT images of more than 1000 patients, selectively chose over 2431 CT images, and created an effective predictive diagnostic system to recognize coronavirus infection using chest CT-Scan images.

The coronavirus infection can be diagnosed in several ways, one of which is laboratory testing (which includes an anti-gen test, an anti-bodies test, a virology test, and imaging of the lung and chest area). Although all these testing methods have varied results, the latter can be

used to automate the diagnosis of the virus using computer tomography (CT) images. There are numerous varieties of deep learning architectures, but not all of them are time-efficient or highly accurate. This article talks about the Densely Connected Neural Network (also known as DenseNet), more specifically the DenseNet-121 architecture [4]–[5]. It is used with a boosting support vector binary classifier to make predictions more often. The model trained with these two methods together (deep learning architecture and machine learning classifier) does a great job of diagnosing CoVID-19 [6]–[7] infections from Chest-CT scan images. This work is made portable by introducing a web API service with the help of Python's micro-web service Flask.

### 1.1 Research Problem

The research aims in addressing the need for a robust framework to diagnose novel coronavirus (COVID-19) infections from CT images efficiently and accurately.

### 1.2 Objectives of the Study

- To develop a ready-to-deploy computer-aided diagnosis (CADx) system using deep learning architectures to detect COVID-19 infection from CT images.
- To experiment with different deep learning architectures, including the Densely Connected Convolution Network (DenseNet-121 architecture), to find the most reliable predictive model for diagnosing COVID-19.
- To achieve high accuracy, precision, F1 score and recall, in diagnosing COVID-19 using the proposed framework.
- To demonstrate the feasibility of using the DenseNet-121 architecture and a boosting support vector binary classifier for distinguishing between COVID-19 positive and healthy individuals.
- To reduce the time required for diagnosing COVID-19 by developing a model that can process one CT image in less than 1 second.
- To provide a web API service using Python's micro-web service Flask to make the diagnostic system portable and accessible.

## 2. Literature Review

Computer vision plays a significant role in autonomous systems that mimic human visual systems in their performance of tasks. In certain situations, computer vision performs better than human vision [8]. With the aid of medical imaging data, computer vision improves the treatment, prediction, and diagnosis of disease [9]. A convolutional neural network is one of the most prominent deep neural network architectures for image classification [10].

Medical radiography is one of the most important methods of diagnosing internal infections. In most of the CoVID-19 infections, during the inceptive stage, the CT images have more or less similar layered pixels. The most important stage in controlling the unfolding of COVID-19 is instantly diagnosing each patient as a healthy individual. Currently, the primarily used method of diagnosis is the RT-PCR test [11]-[14]. On average, an RT-PCR test takes anywhere from 4 to 8 hours to analyse the sample. But on the other hand, radiologists and medical specialists require an average time of ~31 minutes to test the CT images and confirm the infection. Due to a lack of public CT-scan images [15], the current architectures (VGG-16, ResNet-50, InceptionV2, and MobileNet) are not very effective. They also needed a lot of resources (training data, computing power, processing time, etc.) to run the model, as shown in Table 1. The results in the below table were acquired while comparing the trained model with other architectures on the same dataset.

**Table 1.** Time Comparison for Model Evaluation

| Deep Learning Architecture | Time taken for model evaluation |
|---|---|
| ResNet-50 | 625 seconds |
| InceptionV2 | 1096 seconds |
| MobileNet V1 | 780 seconds |
| MobileNet V2 | 567 seconds |
| DenseNet-121 | 338 seconds |

VGG16 is a CNN (Convolutional Neural Network), a type of network. There are sixteen weighted layers, or 16 in VGG16. VGG16 uses an input tensor size of 224,244 with three RGB channels. VGG16 uses convolution layers of 3x3 filters with stride 1 and the same padding and max pool layers of 2x2 filters with stride 2 in place of a huge number of hyper-parameters. Thirteen convolutional layers, five Max Pooling layers, three dense layers, and twenty-one total layers make up VGG16; however, only sixteen weight layers, or learnable parameter layers, are present. There are 64 filter numbers for Conv-1 Layer, 128 filter numbers for Conv-2, 256 filter numbers for Conv-3, and 512 filter numbers for Convs 4 and 5. In Figure 1, we have shown the VGG-16 architecture [5].

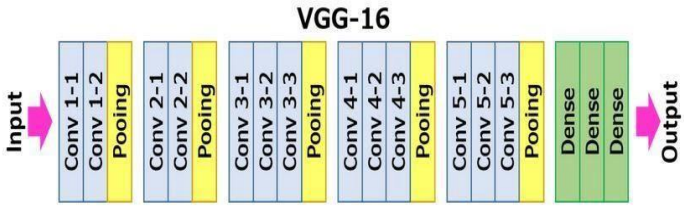


**Figure 1.** VGG-16 Architecture [5]

Compared to VGG16, ResNet has a deeper network, but because global average pooling is used instead of fully connected layers, it is more compact. It aims to overcome the diminishing gradient descent problem by adding some connections straight to the output and removing training from a few layers. This is known as residual networks. This implies that we can use this kind to train incredibly deep networks. This category includes ResNet50, which contains 50 layers. Figure 2, shows the ResNet-50 architecture [5].

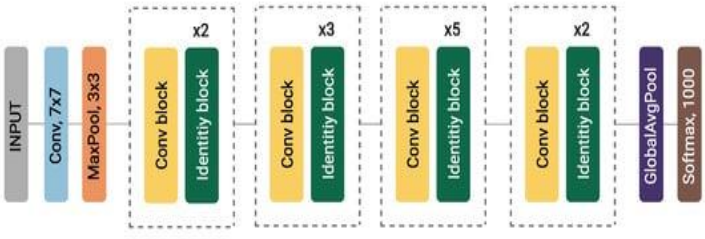


**Figure 2.** ResNet-50 Architecture [5]

Inception is originally referred to as GoogleNet. Although VGG is an effective model, it has a high computational cost in terms of memory and effort. By incorporating a bottleneck layer (1X1 convolutional filter), Inception is able to reduce costs. In addition, it employs

convolutions of various diameters, such as 5X5, 3X3, and 1X1, to capture the finer details. In addition, it reduces the total number of parameters by substituting the fully connected layers with a global average pooling following the final convolutional layer.

VGG is a powerful model, but it requires a lot of memory and computer power. Furthermore, it uses convolutions with different diameters, including 3X3, 5X5, and 1X1, to capture the finer features. Additionally, it reduces the number of parameters by inserting a global average pooling layer after the final convolutional layer in place of the fully connected layers.

The MobileNetV2 architecture improves upon the predecessor's MobileNetV1 design. It maintains depth-separable convolution as an effective construction element and introduces linear bottlenecks and short-cut connections.

## 3. Proposed System

After carefully evaluating the previously available systems, we gathered a vast variety of CT-scan images of more than ~1,40,000 [9] from the github repository with a dataset named "COVID-CS," more than ~4000 labeled (CoVID positive and negative separated) CT-scan images [10] from the Kaggle repository with the “SARS-COV-2 CT-Scan Dataset,” and other datasets from the Harvard Dataverse repository with a dataset named "Covid19-CT-dataset." These images were randomly chosen to train the model for this proposed system with the DenseNet-121 architecture. The training dataset and test dataset was split in the ratio of 80:20 and used for implementation. This architecture was specifically chosen because it has a lower tendency to overfit when compared with other popular architectures.

In Figure 3, represents the proposed approach architecture. The architecture of the proposed system is subdivided into the following modules:

A. Data Pre-processor
B. Model Architecture
C. Model Creation
D. Model Evaluation
E. Bulk Image Tester
F. Standalone Web Service for Public Access

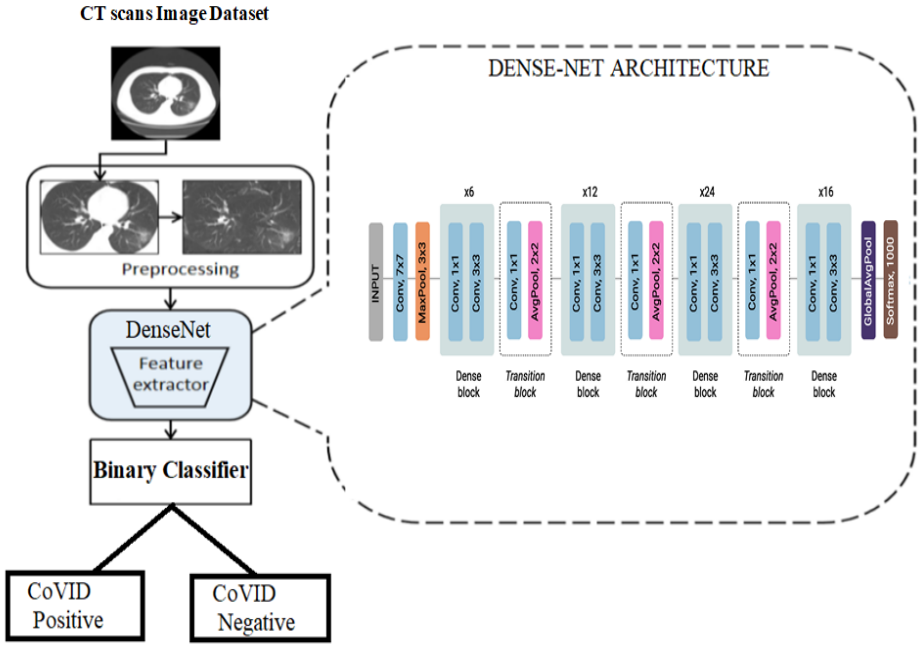


**Figure 3.** Architecture of Proposed System

## A. Data pre-processor

The data pre-processor module primarily aims to normalize the input CT-scan images. The CT-scan images are provided as input in any one of the following formats: JPEG, JPG, or PNG. These images are initially stored in two different directories (the Covid positive directory and the Covid negative directory). The images are fetched from the file system and stored in the format of lists, and respective labels are created while fetching the images. The input images are resized to the dimensions (224, 224, 3), where 3 is the number of color channels (RGB). The images are normalized using the NumPy class by reducing the image values to numerical (float) values. With the help of MATLAB, the normalized (pre-processed) input images are stored in Mat File Format.

## B. Model Architecture

The DenseNet-121 architecture uses a variable number of layers (repetitions) for each dense block, each having two convolutions: a 3x3-sized kernel for the convolution operation and a 1x1-sized kernel for the bottleneck layer. Moreover, a 2x2 average pooling layer with a 2-stride and a 1x1 convolutional layer make up each transition layer.

DenseNet-121 is composed of the following layers: 4 average pooling, 1 fully connected layer, 61 1x1 convolution, 58 3x3 convolution, and 1 7x7 convolution. DenseNet-

121 is made up of four average pooling layers and 120 convolutions. The output quantities for each DenseNet-121 layer are shown in Table 2.

### C. Model Creation

The normalized images stored in the .Mat file are fetched and stored in lists. The model was initially created with a base image dimension of 224, 224, 3 because of the restriction of DenseNet-121. TensorFlow Keras is equipped to simplify the process of model creation through the use of ImageNet weights. In order to reduce the dimension of the image during the model training process (in accordance with the DenseNet-121 architecture specification), the Global Average Pooling 2D class is used. This helps to reduce the image dimension as it transcends through various layers of DenseNet-121. The features of these images are extracted using this method and stored in a separate .Mat file for model evaluation and prediction. In dense block (D) of DenseNet-121, each dense layer (DL) adds 32 new feature maps to the previous volume; consequently, we progress from 64 to 256 through six layers. Transition Block (T) is a 1x1 convolution with 128 filters followed by a 2x2 pooling with a stride of 2, which halves the volume and the number of feature maps.

### D. Feature Extractor

The DenseNet-121 architecture is employed for feature extraction in the training process. Densely connected convolutional layers are found in each of the architecture's several dense blocks. DenseNet-121 extracts both low-level and high-level features from input CT images by learning them through a hierarchical representation. The design uses a pooling operation with a stride of two, which lowers the volume and quantity of feature maps.

**Table 2.** Various Layers and their Output Size in DenseNet-121 Architecture

| Layers | Output Size | DenseNet-121 |
|---|---|---|
| Convolution | 112×112 | 7×7 conv, stride 2 |
| Pooling | 56×56 | 3×3 max pool, stride 2 |
| Dense Block 1 | 56×56 | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$ |
| Transition Layer 1 | 56×56<br>28×28 | 1×1 conv<br>2×2 average pool, stride 2 |
| Dense Block 2 | 28×28 | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$ |
| Transition Layer 2 | 28×28<br>14×14 | 1×1 conv<br>2×2 average pool, stride 2 |
| Dense Block 3 | 7×7 | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$ |
| Transition Layer 3 | 14×14<br>7×7 | 1×1 conv<br>2×2 average pool, stride 2 |
| Dense Block 4 | 7×7 | $\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$ |
| Output | 1×1 | 7×7 global average pool |

## E. Model Evaluation

During model evaluation, a cross-validation of 10 epochs was used to identify the overall performance of the model. With the help of this technique, the normalized (pre-processed) images are randomly divided into ‘N’ adjunct batches, with ‘N-1’ fields considered the training data and the left-over bunch considered the testing data. At the end, the mean of all N-batches presents the model’s performance analysis. The primary advantage of utilizing this technique is that the entire dataset is equipped for model analysis, thereby reducing information flush during the evaluation.

A support vector machine binary classifier is used with this model during evaluation to improve the rate of prediction. The chosen binary classifier was employed in the classification stage because it has been proven to be effective in distinguishing between COVID-19 positive and healthy individuals with high accuracy, recall, precision, and F1 score. When we combine deep learning architectures with a machine learning-based classifier, the model's true positive (TP), false negative (FN), false positive (FP), and true negative (TN) values all get much better. True positive rate (TPR) is also known as sensitivity, and false positive ratio (FPR) is also known as false positive rate. The evaluation metrics (performance measures) are calculated as follows:

$$\text{Sensitivity (TPR)} = TP/(TP+FN) \quad (1)$$

$$\text{False Positive Rate (FPR)} = FP/(TN+FP) \quad (2)$$

$$\text{Recall} = TP/(TP+FN) \quad (3)$$

$$\text{Precision} = TP/(TP+FP) \quad (4)$$

$$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN) \quad (5)$$

$$\text{AUC} = \sum_{i=0}(FPR_{i+1} - FPR_i) * (TPR_1 + TPR_{i+1}) \quad (6)$$

$$\text{F1 Score} = 2 * \left(\frac{precision*recall}{precision+recall}\right) \quad (7)$$

## F. Bulk Image Tester

In the bulk image tester, the picking process (serializing and de-serializing Python object structures) is used to store the model checkpoint for external image testing. A support vector machine binary classifier is integrated into this checkpoint so as to improve the overall prediction rate. The input images (testing images) are normalized, resized (to the dimension of 224 x 224 x 3) using the NumPy class, and predicted using the model checkpoint created above. The image tester is capable of testing images in the following formats: JPEG, JPG, and PNG. A separate directory to place the testing images is required to be created in the local system for this purpose. In order to test all the images placed in the testing directory, Python's standard library (Glob) is used to select the image using the pattern matching technique. This study says that the saliency-guided morphology-aware U-Net (SMU-Net) works better than some other

cutting-edge deep-learning algorithms when it comes to separating breast lesions in ultrasound images.

### G. Standalone Web Service for Public Access

In this research, Python's micro web service framework (Flask) is equipped to enable a standalone web service for external testing of Chest-CT scan images. The service is implemented in a modern way, such that the scan image is docked to the application through Hyper Text Mark-up Language (HTML). Concurrently, the docked image is compared with the model checkpoint, and the analysis is presented to the user on the browser as shown in Figure 4.

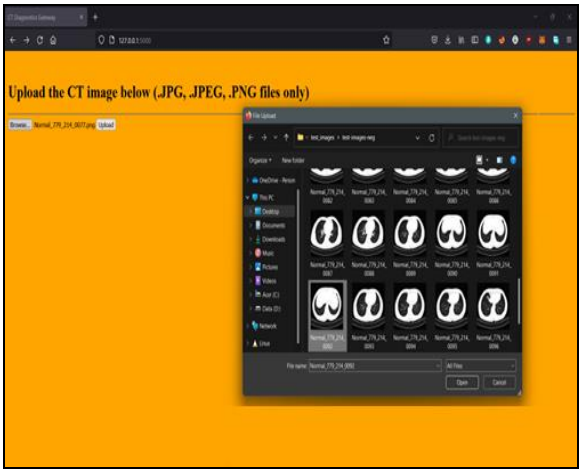


**Figure 4.** Web Page for Public Image Testing

### H. Criteria for Selecting the Models for Feature Extraction and Classification

The models for feature extraction and classification was selected based on their performance in accurately diagnosing COVID-19 from CT images. The DenseNet-121 architecture was chosen for feature extraction as it has been widely used in image classification tasks and has shown promising results in medical imaging applications. The boosting support vector binary classifier was employed for classification due to its effectiveness in distinguishing between COVID-19 positive and healthy individuals with high accuracy, recall, precision, and F1 score. The combination of DenseNet-121 and the binary classifier algorithm achieved an accuracy of 93% ± 1.8%, recall of 94.9% ± 2.6%, precision of 98% ± 1.5%, and an F1 score of 94% ± 1.7%, demonstrating their suitability for the task of COVID-19 diagnosis. The selected models were able to process CT images in less than 1 second, making them

efficient for real-time diagnosis. The models were chosen based on their ability to provide reliable and accurate predictions, enabling prompt discovery of COVID-19 infections and assisting in controlling the spread of the virus.

## 4. Results

The results to demonstrate the classification of the Corona virus is presented in this section. Experimentations are performed in Python, and for convolutional neural networks cv2, flask, flask_ngrok, numpy, pickle, pil, scipy.io, tensorflow, sklearn.svm, sklearn.model_selection, time, and warnings, Python dependencies are installed using the command >> pip install <>. The software required for this experimentation is Python 3.2 and above, Python IDLE, and Windows 8 and above.

The dataset used in the experiment is obtained and combined from the github repository titled "COVID-CS," the Kaggle repository titled "SARS-COV-2 CT-Scan Dataset," and the Harvard Dataverse repository titled "Covid-19-CT-dataset." More than 1,000,000 chest CT images from 1,000+ patients with confirmed COVID-19 infections constitute the dataset. The age distribution of patients who underwent CT imaging was $47.18 \pm 16.32$ (mean $\pm$ standard deviation) years, and the age range was between 6 and 89 years. Board-certified radiologists reviewed each CT image in these libraries to check for COVID-19 infections. In the event that the initial two radiologists couldn't agree, a third, more seasoned radiologist made the ultimate determination on the presence of COVID-19 infections.

During this work, a diverse group of deep learning architectures were compared alongside the support vector machine classifier. Python performs all of the processing work in conjunction with dependencies like Scîkit-learn, TensorFlow, Pickle, Keras, Sklearn, NumPy, etc. and architectures like ResNet-50, MobileNe-V1, MobileNet-V2, and Inception-V3 after rescaling the images to extract features. These features are appended to the support vector machine to analyse the image prediction.

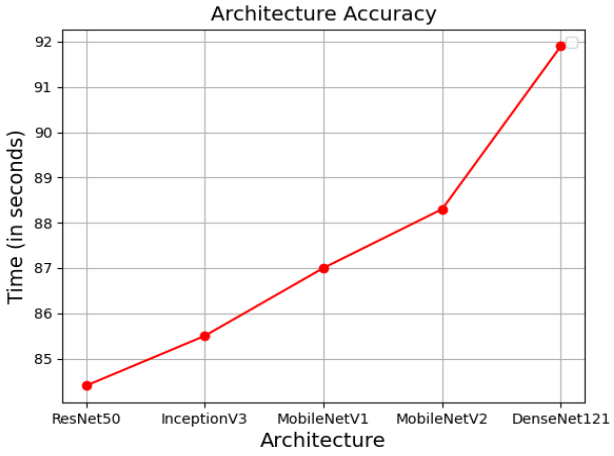


**Figure 5.** Comparison of Accuracy Among Architectures

In Figure 5, a visualization of the comparison of accuracy among the above-mentioned architectures is presented. From this, we can conclude that DenseNet-121 has the best accuracy for the collected dataset.

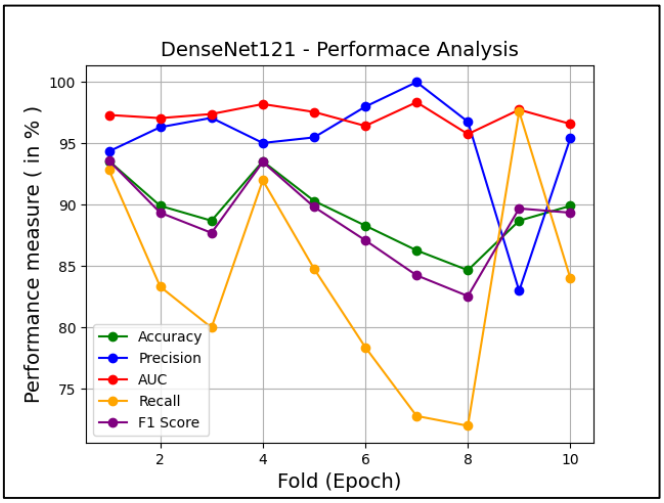


**Figure 6.** Performance Analysis of DenseNet-121

Figure 6 represents the 10-epoch performance metric for the trained model. It can be observed that the model misclassified 8.2% of infected images as healthy conditions and 0.5% of normal images as infected images.

**Table 3.** Comparison of Accuracy among Architectures

| Architecture | Time Taken (seconds) | Accuracy (%) | Auc (%) | Recall (%) | Precision (%) | F1-Score (%) |
|---|---|---|---|---|---|---|
| ResNet-50 | 625 | 84.4 | 81.9 | 75.3 | 90.3 | 85.2 |
| Inception-V3 | 1096 | 85.5 | 97 | 91.5 | 91.6 | 91.6 |
| MobileNet-V1 | 780 | 87 | 94.1 | 86 | 90.1 | 89.5 |
| MobileNet-V2 | 567 | 88.3 | 97.9 | 90 | 98.9 | 90.5 |
| DenseNet-121 | 338 | 92.1 | 98.1 | 91.8 | 99.5 | 92.3 |

A comparison of the accuracy of different architectures is shown in Table 3. It shows the time it took for each architecture to achieve a certain accuracy level (AUC) in recall precision and the percentage of correct assessments (%).

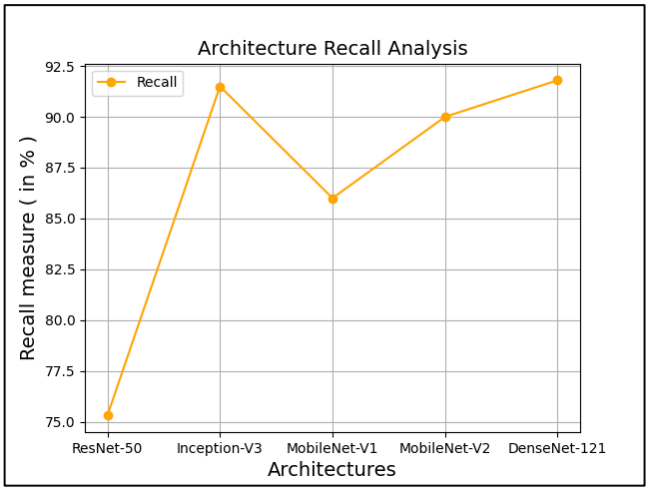


**Figure 7.** Comparison of Recall between Architectures

From the above comparison shown in Figure 7, it can be inferred that the classifier is more indulgent in classifying something as positive.

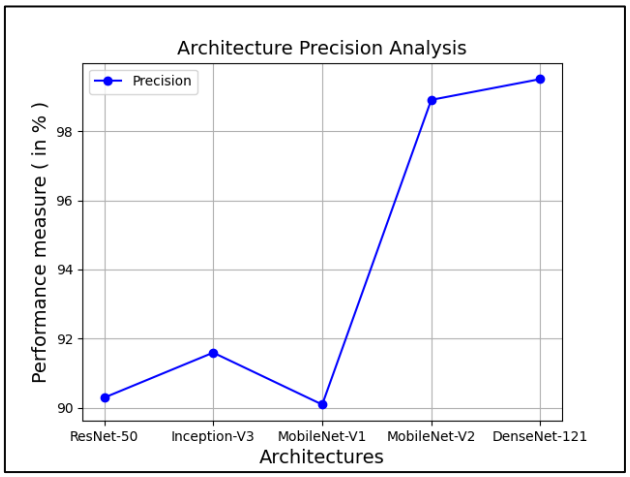


**Figure 8.** Comparison of Precision between Architectures

From the above comparison shown in Figure 8, it can be inferred that the model has superior precision over other architectures, thereby proving to be consistent. The computed incongruous mean of precision and recall is often called the F beta score (F1-score). The model by far created using the DenseNet-121 architecture achieved an F1-score of 92.3%.

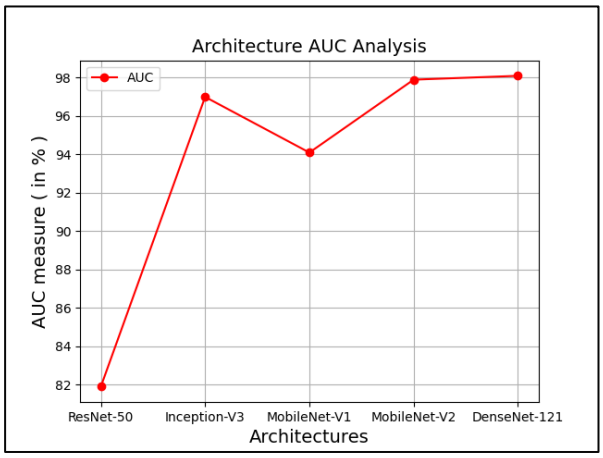


**Figure 9.** Comparison of Area under Curve (AUC) among Architectures

In Figure 9, AUC describes the overall performance of the trained model. It can be inferred that the model is better if the AUC is greater.

## 5. Conclusion and Future Scope

The DenseNet-121 architecture utilized in this work comparatively has fewer trainable parameters (by extracting features from images) and hence results in faster processing of the model. Support Vector Machine (Nu-SVM) and the deep learning architecture DenseNet-121

worked together to get better results. They did a 10-epoch classification in 338 seconds, which is 53.3% faster than the average of the architectures we looked at earlier. They achieved an accuracy of 92.1%, an AUC of 98.1%, precision of 99.5%, recall of 91.8%, and an F1 score of 92.3%. On a side note, the proposed system is also facilitated by a feature to be hosted on a public domain service, thereby allowing general public (or) medical supervisors to test the CT-scan images independently.

Using deep learning optimization algorithms like stochastic gradient descent (SGD), adaptive gradient descent (AdaGrad), and others can enhance the application's Web service functionality. These can make the training models better, which can lead to a better prediction of infection.

## References

[1] Team, Epidemiology. "The epidemiological characteristics of an outbreak of 2019 novel coronavirus diseases (COVID-19)—China, 2020." China CDC weekly 2, no. 8 (2020): 113.

[2] Mohammadpoor, Mojtaba. "A deep learning algorithm to detect coronavirus (COVID-19) disease using CT images." PeerJ Computer Science 7 (2021): e345.

[3] Huang, Thomas S. "Computer vision: Evolution and promise." CERN European Organization for Nuclear Research-Reports-CERN (1996): 21-26.

[4] Serena Low, Woan Ching, Joon Huang Chuah, Clarence Augustine TH Tee, Shazia Anis, Muhammad Ali Shoaib, Amir Faisal, Azira Khalil, and Khin Wee Lai. "An overview of deep learning techniques on chest X-ray and CT scan identification of COVID-19." Computational and Mathematical Methods in Medicine 2021 (2021): 1-17.

[5] Wu, Yu-Huan, Shang-Hua Gao, Jie Mei, Jun Xu, Deng-Ping Fan, Rong-Guo Zhang, and Ming-Ming Cheng. "Jcs: An explainable covid-19 diagnosis system by joint classification and segmentation." IEEE Transactions on Image Processing 30 (2021): 3113-3126.

[6] Soares, Eduardo, Plamen Angelov, Sarah Biaso, Michele Higa Froes, and Daniel Kanda Abe. "SARS-CoV-2 CT-scan dataset: A large dataset of real patients CT scans for SARS-CoV-2 identification." MedRxiv (2020): 2020-04.

[7] Fang, Yicheng, Huangqi Zhang, Jicheng Xie, Minjie Lin, Lingjun Ying, Peipei Pang, and Wenbin Ji. "Sensitivity of chest CT for COVID-19: comparison to RT-PCR." Radiology 296, no. 2 (2020): E115-E117.

[8] Chen, Chi Hau. "An introduction to computer vision in medical imaging." In Computer Vision in Medical Imaging, pp. 1-16. 2014..

[9] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks." Communications of the ACM 60, no. 6 (2017): 84-90.

[10] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[11] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. "Mobilenetv2: Inverted residuals and linear bottlenecks." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 4510-4520. 2018.

[12] Mohammadpoor, Mojtaba. "A deep learning algorithm to detect coronavirus (COVID-19) disease using CT images." PeerJ Computer Science 7 (2021): e345.

[13] Udugama, Buddhisha, Pranav Kadhiresan, Hannah N. Kozlowski, Ayden Malekjahani, Matthew Osborne, Vanessa YC Li, Hongmin Chen, Samira Mubareka, Jonathan B. Gubbay, and Warren CW Chan. "Diagnosing COVID-19: the disease and tools for detection." ACS nano 14, no. 4 (2020): 3822-3835.

[14] Zhang, N., D. Lei, and J. F. Zhao. "An improved Adagrad gradient descent optimization algorithm." In 2018 Chinese Automation Congress (CAC), pp. 2359-2362. IEEE, 2018.

[15] Li, Xi-Lin. "Preconditioned stochastic gradient descent." IEEE transactions on neural networks and learning systems 29, no. 5 (2017): 1454-1466.

[16] Islam, Md Mohaiminul, Tanveer Hannan, Laboni Sarker, and Zakaria Ahmed. "COVID-DenseNet: a deep learning architecture to detect COVID-19 from chest radiology images." In Proceedings of International Conference on Data Science and Applications: ICDSA 2022, Volume 2, pp. 397-415. Singapore: Springer Nature Singapore, 2023.

## Author's biography

**Alok Kumar** is working as an Assistant Professor in the Department of Computer Science and Engineering at Government College of Engineering, Dharmapuri, Tamilnadu. His research interest includes Knowledge in Data Mining, Machine Learning, Deep Learning, Computer Vision.

**Dr. N. Mahendran** is working as an Associate Professor in the Department of Electronics and Communication Engineering at M. Kumarasamy College of Engineering, Karur, Tamilnadu. His research interest includes Knowledge in Machine Learning, Deep Learning, Computer Vision.